LIST OF CLAIMS:

This listing of claims will replace all prior versions and listing of claims in the above-referenced application.

1. (original) A method for detecting an inconsistent data structure comprising:

receiving a specification describing at least one consistency constraint of a data structure; and

dynamically determining during execution of a program whether said data structure violates said at least one consistency constraint.

2. (original) The method of Claim 1, wherein said specification comprises at least one logical formula.

3. (original) The method of Claim 2, wherein said specification includes at least one consistency constraint expressed in terms of said data structure.

4. (original) The method of Claim 3, wherein, prior to dynamically determining whether said data structure violates said at least one consistency constraint, it is determined whether repairing the data structure according to the at least one consistency constraint will terminate.

5. (original) The method of Claim 3, wherein said specification includes a description of said data structure.

6. (original) The method of Claim 4, wherein said specification includes a description of said data structure.

7. (original) The method of Claim 1, further comprising:

representing said data structure as an abstract model; and

determining consistency constraint violations of said abstract model.

8. (original) The method of Claim 7, wherein said specification includes a description of said data structure.

9. (original) The method of Claim 8, wherein said specification includes an abstract model definition.

10. (original) The method of Claim 9, wherein said specification includes an internal constraint in terms of said abstract model definition.

11. (original) The method of Claim 10, further comprising:

determining if said internal constraint is violated in accordance with an evaluation of said internal constraint.

12. (original) The method of Claim 11, wherein said specification includes at least one external constraint mapping elements of said abstract model to elements of said data structure.

13. (original) The method of Claim 10, wherein said description of said abstract model includes at least one model definition rule and at least one declaration for one of: a set and a relation, said at least one model definition rule representing an element of said data structure in at least one of a set and a relation.

14. (original) The method of Claim 13, wherein said specification includes at least one external constraint mapping elements of said abstract model to elements of said data structure.

15. (original) The method of Claim 1, wherein said dynamically determining is performed in response to at least one of: an explicit call and a transfer of control to an error handler.

16. (original) The method of Claim 13, wherein, prior to dynamically determining whether said data structure violates said at least one consistency constraint, it is determined whether construction of said abstract model will terminate.

17. (original) The method of Claim 16, wherein, prior to dynamically determining whether said data structure violates said at least one consistency constraint, it is determined whether said at least one model definition rule has cyclic dependencies which involve negation operators.

18. (original) The method of Claim 17, wherein said at least one model definition rule is of the form: quantifier, Q, guard, G, and an inclusion constraint, I, and the method further comprising:

translating each guard of each of said at least one model definition rule into disjunction normal form including a logical ORing of conjunctions, each of said conjunctions including one or more predicates;

constructing a graph representing said at least one model definition rule, said graph including a node for each model definition rule, a normal edge from a first rule to a second rule if the inclusion constraint for the first rule uses a set or relation which is also used in a guard of the second rule or a quantifier of the second rule, a negated edge from the first rule to the second rule if the inclusion constraint for the first rule uses a set or a relation which is negated in connection with one of a set or relation of the second rule's guard; and

determining if there are any cycles in said graph with negated edges.

19. (original) The method of Claim 8, wherein, prior to dynamically determining whether said data structure violates said at least one consistency constraint, it is determined whether repairing said internal constraints will terminate.

20. (original) The method of Claim 1, further comprising:

determining whether a memory reference in connection with said data structure is valid in accordance with currently allocated memory of said program.

21. (original) The method of Claim 1, further comprising:

repairing said data structure if said data structure violates said at least one consistency constraint.

22. (original) A method of dynamically repairing an inconsistent data structure during program execution comprising:

receiving at least one inconsistency violation;

selecting a repair to correct said at least one inconsistency violation; and

repairing said inconsistent data structure.

23. (original) The method of Claim 22, further comprising:

resuming execution of said program.

24. (original) The method of Claim 22, further comprising:

performing said repair and satisfying said consistency constraint.

25. (original) The method of Claim 22, wherein said inconsistent data structure is represented in an abstract model, and the method comprising:

repairing said abstract model in accordance with an internal consistency constraint; and

applying a repair to the inconsistent data structure in accordance with an external constraint translating said repair from said abstract model to said inconsistent data structure.

26. (original) The method of Claim 22, further comprising:

repairing said inconsistent data structure in accordance with an internal consistency constraint.

27. (original) The method of Claim 22, further comprising:

selecting a repair from a plurality of repairs in accordance with a cost associated with each repair.

28. (original) The method of Claim 27, wherein said cost is user specified.

29. (original) The method of Claim 27, wherein said inconsistency violation includes a plurality of conditions, and the method further comprising:

determining which of said plurality of conditions are true; and

determining a cost for repairing said inconsistency violation in accordance with those conditions that are not true.

30. (original) A method of handling an invalid memory reference comprising:

determining whether a memory reference associated with an operation is invalid; and

if said memory reference is invalid, performing a substitute action selected in accordance with said operation in place of performing said operation.

31. (original) The method of Claim 30, further comprising:

if said memory reference is associated with a read operation, supplying a default value as a result of performing said read operation; and

if said memory reference is associated with a write operation, disregarding said write operation.

32. (original) The method of Claim 31, wherein at least one invalid read operation has a different default value than at least one other invalid read operation.

33. (original) The method of Claim 30, wherein said invalid memory access is determined during execution of said program.

34. (original) The method of Claim 31, wherein said determining is performed in accordance with memory allocations associated with a program execution.

35. (original) The method of Claim 34, further comprising:

evaluating said memory reference prior to attempting to access a portion of memory.

36. (original) The method of Claim 35, wherein at least one of said read operation and said write operation uses one of: a pointer access, and an array element for said memory reference.

37. (original) The method of Claim 36, wherein a program having an invalid memory reference continues execution following execution of said substitute action.

38. (original) The method of Claim 32, wherein a program having an invalid memory reference continues execution following execution of said substitute action.

39. (original) A computer program product that detects an inconsistent data structure comprising executable code that:

receives a specification describing at least one consistency constraint of a data structure; and

dynamically determines during execution of a program whether said data structure violates said at least one consistency constraint.

40. (original) The computer program product of Claim 39, wherein said specification comprises at least one logical formula.

41. (original) The computer program product of Claim 40, wherein said specification includes at least one consistency constraint expressed in terms of said data structure.

42. (original) The computer program product of Claim 41, further comprising executable code that, prior to dynamically determining whether said data structure violates said at least one consistency constraint, determines whether repairing the data structure according to the at least one consistency constraint will terminate.

43. (original) The computer program product of Claim 41, wherein said specification includes a description of said data structure.

44. (original) The computer program product of Claim 42, wherein said specification includes a description of said data structure.

45. (original) The computer program product of Claim 39, further comprising executable code that:

represents said data structure as an abstract model; and

determines consistency constraint violations of said abstract model.

46. (original) The computer program product of Claim 45, wherein said specification includes a description of said data structure.

47. (original) The computer program product of Claim 46, wherein said specification includes an abstract model definition.

48. (original) The computer program product of Claim 47, wherein said specification includes an internal constraint in terms of said abstract model definition.

49. (original) The computer program product of Claim 48, further comprising executable code that:

determines if said internal constraint is violated in accordance with an evaluation of said internal constraint.

50. (original) The computer program product of Claim 49, wherein said specification includes at least one external constraint mapping elements of said abstract model to elements of said data structure.

51. (original) The computer program product of Claim 48, wherein said description of said abstract model includes at least one model definition rule and at least one declaration for one of: a set and a relation, said at least one model definition rule representing an element of said data structure in at least one of a set and a relation.

52. (original) The computer program product of Claim 51, wherein said specification includes at least one external constraint mapping elements of said abstract model to elements of said data structure.

53. (original) The computer program product of Claim 39, wherein said executable code that dynamically determines is responsive to at least one of: an explicit call and a transfer of control to an error handler.

54. (original) The computer program product of Claim 51, further comprising executable code that, prior to dynamically determining whether said data structure violates said at least one consistency constraint, determines whether construction of said abstract model will terminate.

55. (original) The computer program product of Claim 54, further comprising executable code that, prior to dynamically determining whether said data structure violates said at least one consistency constraint, determines whether said at least one model definition rule has cyclic dependencies which involve negation operators.

56. (original) The computer program product of Claim 55, wherein said at least one model definition rule is of the form: quantifier, Q, guard, G, and an inclusion constraint, I, and the computer program product further comprising executable code that:

translates each guard of each of said at least one model definition rule into disjunction normal form including a logical ORing of conjunctions, each of said conjunctions including one or more predicates;

constructs a graph representing said at least one model definition rule, said graph including a node for each model definition rule, a normal edge from a first rule to a second rule if the inclusion constraint for the first rule uses a set or relation which is also used in a guard of the second rule or a quantifier of the second rule, a negated edge from the first rule to the second rule if the inclusion constraint for the first rule uses a set or a relation which is negated in connection with one of a set or relation of the second rule's guard; and

determines if there are any cycles in said graph with negated edges.

57. (original) The computer program product of Claim 46, further comprising executable code that, prior to dynamically determining whether said data structure violates said at least one consistency constraint, determines whether repairing said internal constraints will terminate.

58. (original) The computer program product of Claim 39, further comprising executable code that:

determines whether a memory reference in connection with said data structure is valid in accordance with currently allocated memory of said program.

59. (original) The computer program product of Claim 39, further comprising executable code that:

repairs said data structure if said data structure violates said at least one consistency constraint.

60. (original) A computer program product that dynamically repairs an inconsistent data structure during program execution comprising executable code that:

receives at least one inconsistency violation;

selects a repair to correct said at least one inconsistency violation; and

repairs said inconsistent data structure.

61. (original) The computer program product of Claim 60, further comprising executable code that:

resumes execution of said program.

62. (original) The computer program product of Claim 60, further comprising executable code that performs said repair and satisfies said at least one consistency constraint.

63. (original) The computer program product of Claim 60, wherein said inconsistent data structure is represented in an abstract model, and the computer program product comprising executable code that:

repairs said abstract model in accordance with an internal consistency constraint; and

applies a repair to the inconsistent data structure in accordance with an external constraint translating said repair from said abstract model to said inconsistent data structure.

64. (original) The computer program product of Claim 60, further comprising executable code that:

repairs said inconsistent data structure in accordance with an internal consistency constraint.

65. (original) The computer program product of Claim 60, further comprising executable code that:

selects a repair from a plurality of repairs in accordance with a cost associated with each repair.

66. (original) The computer program product of Claim 65, wherein said cost is user specified.

67. (original) The computer program product of Claim 65, wherein said inconsistency violation includes a plurality of conditions, and the computer program product further comprising executable code that:

determines which of said plurality of conditions are true; and

determines a cost for repairing said inconsistency violation in accordance with those conditions that are not true.

68. (original) A computer program product that handles an invalid memory reference comprising executable code that:

determines whether a memory reference associated with an operation is invalid; and

if said memory reference is invalid, performs a substitute action selected in accordance with said operation in place of performing said operation.

69. (original) The computer program product of Claim 68, further comprising executable code that:

if said memory reference is associated with a read operation, supplies a default value as a result of performing said read operation; and

if said memory reference is associated with a write operation, disregards said write operation.

70. (original) The computer program product of Claim 69, wherein at least one invalid read operation has a different default value than at least one other invalid read operation.

71. (original) The computer program product of Claim 68, wherein said invalid memory access is determined during execution of said program.

72. (original) The computer program product of Claim 69, wherein said executable code that determines is performed in accordance with memory allocations associated with a program execution.

73. (original) The computer program product of Claim 72, further comprising executable code that:

evaluates said memory reference prior to attempting to access a portion of memory.

74. (original) The computer program product of Claim 73, wherein at least one of said read operation and said write operation uses one of: a pointer access, and an array element for said memory reference.

75. (original) The computer program product of Claim 73, wherein a program having an invalid memory reference continues execution following execution of said substitute action.

76. (original) The computer program product of Claim 70, wherein a program having an invalid memory reference continues execution following execution of said substitute action.